TEC | Tecnológico de Costa Rica

Tarea 4

Introducción y Taller de programación
I-Semestre 2022

Estudiante Marco Rodríguez
Estudiante Maximilian Latysh

Prof. Edgar Rojas

7 de abril de 2022

1. Operaciones Básicas

1.1. mcm(a, b)

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Vamos a estar iterando los valores de d desde 1 hasta a
Dominio: Todos los valores son numeros naturales mayores que cero
Codomino: La salida es un numero natural mayor que cero
'''
def mcmAux1(a, b, d, mcm):
    if d > a or mcm < a*b:
        return mcm
    return mcmAux1(a, b, d+1, mcmAux2(a, b,d, 1, mcm))
'''
Explicacion: Iterando por los valores de c busque el caso de a por c es
    igual a b por d
Dominio: Todos los valores son numeros naturales mayores que cero
Codomino: La salida es un numero natural mayor que cero
'''
def mcmAux2(a, b, d, c, mcm):
    if c>b or mcm < a*b:
        return mcm
    if a*c==b*d:
        return mcmAux2(a, b, d, c+1, a*c)
    return mcmAux2(a, b, d, c+1, mcm)
'''
Explicacion: Retorna el minimo comun multiplo de a y b
Dominio: Todos los valores son numeros naturales mayores que cero
Codomino: La salida es un numero natural mayor que cero
'''
def mcm(a, b):
    return mcmAux1(a, b, 1, a*b)
```

1.2. MCD (a,b)

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Vamos a estar iterando por los numeros menores que el
    minimo de a y b desde 2 viendo si el valor es factor de a y b
Dominio: Todos los valores son numeros naturales mayores que cero
Codomino: La salida es un numero natural mayor que cero
'''
def MCDaux(a, b, factor , factor_previo):
    if factor > min(a,b):
        return factor_previo
    if a%factor==0 and b%factor==0:
        return MCDaux(a,b,factor+1,factor)
    return MCDaux(a,b,factor+1,factor_previo)
'''
Explicacion: Encuentra el maximo comun divisor de a y b
Dominio: Todos los valores son numeros naturales mayores que cero
Codomino: La salida es un numero natural mayor que cero
'''
def MCD(a,b):
    return MCDaux(a,b,2,1)
```

1.3. Es-Primo (n)

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Pasamos por todos los numeros impares verificando si son
    factores de n.
# - Si no encontramos una k antes de que se cumpla que k*k es mayor que
    n,
# - paramos el programa.
Dominio: Todos los valores de entrada son numeros naturales mayores que
    2.
Codomino: La salida es True o Falso.
'''
def esPrimoAux(n, k):
   if k*k > n:
      return True
   if n % k == 0:
      return False
   return esPrimoAux(n, k + 2)
'''
Explicacion: Determina si un numero es primo o no.
Dominio: La entrada es un numero natural mayor que 1.
Codomino: La salida es True o Falso.
'''
def esPrimo(n):
   if n == 2:
      return True
   if n % 2 == 0:
      return False
   return espPimoAux(n, 3)
```

1.4. n-simo-primo (n)

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Pasamos por todos los numeros impares verificando si son
    factores de n.
# - Si no encontramos una k antes de que se cumpla que k*k es mayor que
    n,
# - paramos el programa.
Dominio: Todos los valores de entrada son numeros naturales mayores que
    2.
Codomino: La salida es True o Falso.
'''
def esPrimoAux(n, k):
   if k*k > n: return True
   if n % k == 0: return False
   return esPrimoAux(n, k + 2)
'''
Explicacion: Determina si un numero es primo o no.
Dominio: La entrada es un numero natural mayor que 1.
Codomino: La salida es True o Falso.
'''
def esPrimo(n):
   if n == 2: return True
   if n % 2 == 0: return False
   return espPimoAux(n, 3)
'''
Explicacion: Retorna el n-simo numero primo.
Dominio: La entrada es un numero natural mayor que 0.
Rango: La salida son los numeros primos.
'''
def n_simo_primo(num):
    if num==1:
        return 2
    return aux_n_simo_primo(5, num-2)
'''
Explicacion: Cada vez que determina que un numero pos es primo baja num
    hasta que el mismo sea igual a cero
Dominio: La entrada es un numero natural mayor que 0.
Rango: La salida son los numeros primos.
'''
def aux_n_simo_primo(pos, num):
    if num==0:
        return pos-2
    if esprimo(pos):
        return aux_n_simo_primo(pos+2, num-1)
    return aux_n_simo_primo(pos+2,num)
```

2. Sucesiones

2.1. Sumatoria 1

2.1.1. Versión pila

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Iteramos por todos los valores de i menores o iguales que
    1500000 acumulando 2*i.
Dominio: Numeros naturales mayor que cero.
Codominio: La salida son los numeros naturales.
'''
def suma_1_pila(i=1):
   if i > 1500000:
      return 0
   return 2*i + suma_1_pila(i + 1)
```

2.1.2. Versión cola

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Calcule la suma de los numeros pares con los limites
    respectivos.
Dominio: Vacio.
Codominio: La salida es un numero natural.
'''
def suma_1_cola():
   return suma_1_cola_aux(1, 0)
'''
Explicacion: Iteramos por todos los valores de i menores o iguales que
    1500000 acumulando 2*i.
Dominio: Numeros naturales mayor que cero.
Codominio: La salida son los numeros naturales.
'''
def suma_1_cola_aux(i, suma):
   if i > 1500000:
      return suma
   return suma_1_cola_aux(i + 1, suma + 2*i)
```

2.2. Sumatoria 2

2.2.1. Versión pila

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Iteramos por todos los valores de i menores o iguales que
    500000 acumulando 2*i + 1.
Dominio: Numeros naturales mayor que cero.
Codominio: La salida son los numeros naturales.
'''
def suma_2_pila(i=0):
   if i > 500000:
      return 0
   return 2*i + 1 + suma_2_pila(i + 1)
```

2.2.2. Versión cola

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Calcule la suma de los numeros pares con los limites
    respectivos.
Dominio: Vacio.
Codominio: La salida es un numero natural.
'''
def suma_2_cola():
   return suma_2_cola_aux(0, 0)
'''
Explicacion: Iteramos por todos los valores de i menores o iguales que
    500000 acumulando 2*i + 1.
Dominio: Numeros naturales mayor que cero.
Codominio: La salida son los numeros naturales.
'''
def suma_2_cola_aux(i, suma):
   if i > 500000:
      return suma
   return suma_2_cola_aux(i + 1, suma + 2*i + 1)
```

2.3. Sumatoria 3

2.3.1. Versión pila

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Iteramos por todos los valores de i menores o iguales que
    1000 acumulando i*i.
Dominio: Numeros naturales mayor que cero.
Codominio: La salida son los numeros naturales.
'''
def suma_3_pila(i=1):
   if i > 1000:
      return 0
   return i*i + suma_3_pila(i + 1)
```

2.3.2. Versión cola

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Calcule la suma de los numeros pares con los limites
    respectivos.
Dominio: Vacio.
Codominio: La salida es un numero natural.
'''
def suma_3_cola():
   return suma_3_cola_aux(1, 0)
'''
Explicacion: Iteramos por todos los valores de i menores o iguales que
    1000 acumulando i*i.
Dominio: Numeros naturales mayor que cero.
Codominio: La salida son los numeros naturales.
'''
def suma_3_cola_aux(i, suma):
   if i > 1000:
      return suma
   return suma_3_cola_aux(i + 1, suma + i*i)
```

2.4. Sumatoria 4

2.4.1. Versión pila

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Iteramos por todos los valores de i menores o iguales que
    15 acumulando 1/3**i.
Dominio: Numeros naturales mayor que cero.
Codominio: La salida son los numeros naturales.
'''
def suma_4_pila(i=0):
   if i > 15:
      return 0
   return 1/(3**i) + suma_4_pila(i + 1)
```

2.4.2. Versión cola

```
# Ayuda con el limite recursivo
from sys import setrecursionlimit as s
s((1 << 31) - 1)
'''
Explicacion: Calcule la suma de los numeros pares con los limites
    respectivos.
Dominio: Vacio.
Codominio: La salida es un numero natural.
'''
def suma_4_cola():
   return suma_4_cola_aux(0, 0)
'''
Explicacion: Iteramos por todos los valores de i menores o iguales que
    15 acumulando 1/3**i.
Dominio: Numeros naturales mayor que cero.
Codominio: La salida son los numeros naturales.
'''
def suma_4_cola_aux(i, suma):
   if i > 15:
      return suma
   return suma_4_cola_aux(i + 1, suma + 1/(3**i))
```

3. Relaciones de recurrencia

3.1. Patrón 1

3.1.1. Funciones matemáticas

- Función de pila (grado 3):

$$pila(n) = \begin{cases} 1 & si \quad n < 3 \\ \\ pila(n-1) + pila(n-2) + pila(n-3) & sino \end{cases}$$

- Funciones de cola:

$$cola(n) = \begin{cases} 1 & si \quad n < 3 \\ \\ aux(n-3, 1, 1, 1) & sino \end{cases}$$

$$aux(n, x, y, z) = \begin{cases} x + y + z & si \quad n = 0 \\ \\ aux(n-1, y, z, x + y + z) & sino \end{cases}$$

3.1.2. Código

- De pila

```
'''
Explicacio'n: Calcula con recursio'n de pila el n-simo nu'mero del
    patro'n: 1,1,1,3,5,9,17,31,57...
Dominio: Un nu'mero natural (el valor ma's peque~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron1pila(n):
    if n<3:
        return 1
    return patron1pila(n-1)+patron1pila(n-2)+patron1pila(n-3)
```

- De cola

```
'''
Explicacio'n: Calcula el n-simo nu'mero del patro'n:
    1,1,1,3,5,9,17,31,57...
Dominio: Un nu'mero natural (el valor ma's peque~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron1cola(n):
    if n<3:
        return 1
    return patron1aux(n-3,1,1,1)
'''
Explicacio'n: Calcula con recursio'n de cola el n-simo nu'mero del
    patro'n: 1,1,1,3,5,9,17,31,57...
Dominio: 4 nu'meros naturales (el valor ma's pequen~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron1aux(n,x,y,z):
    if n==0:
        return x+y+z
    return patron1aux(n-1,y,z,x+y+z)
```

3.2. Patrón 2

3.2.1. Funciones matemáticas

- Función de pila (grado 1):

$$pila(n) = \begin{cases} 3 & si \quad n = 0 \\ pila(n-1) * 2 & sino \end{cases}$$

- Funciones de cola:

$$cola(n) = aux(n, 3)$$

$$aux(n, mult) = \begin{cases} mult & si \quad n = 0 \\ aux(n-1, mult * 2) & sino \end{cases}$$

3.2.2. Código

- De pila

```
'''
Explicacio'n: Calcula con recursio'n de pila el n-simo nu'mero del
    patro'n: 3,6,12,24,48,96...
Dominio: Un nu'mero natural (el valor ma's peque~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron2pila(n):
    if n==0:
        return 3
    return (patron2pila(n-1)<<1)
```

- De cola

```
'''
Explicacio'n: Calcula el n-simo nu'mero del patro'n:
    3,6,12,24,48,96...
Dominio: Un nu'mero natural (el valor ma's peque~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron2cola(n):
    return patron2aux(n,3)
'''
Explicacio'n: Calcula con recursio'n de cola el n-simo nu'mero del
    patro'n: 3,6,12,24,48,96...
Dominio: 2 nu'meros naturales (el valor ma's pequen~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron2aux(n,mult):
    if n==0:
        return mult
    return patron2aux(n-1,(mult<<1))
```

3.3. Patrón 3

3.3.1. Funciones matemáticas

- Función de pila (grado 2):

$$pila(n) = \begin{cases} 1 & si \quad n = 0 \\ 3 & si \quad n = 1 \\ pila(n-1) * 2 + pila(n-2) & sino \end{cases}$$

- Funciones de cola:

$$cola(n) = \begin{cases} 1 & si \quad n = 0 \\ 3 & si \quad n = 1 \\ aux(n-2, 1, 3) & sino \end{cases}$$

$$aux(n, x, y) = \begin{cases} y * 2 + x & si \quad n = 0 \\ aux(n-1, y, y * 2 + x) & sino \end{cases}$$

3.3.2. Código

- De pila

```
'''
Explicacio'n: Calcula con recursio'n de pila el n-simo nu'mero del
    patro'n: 1,3,7,17,41,99,239,577,1393...
Dominio: Un nu'mero natural (el valor ma's peque~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron3pila(n):
    if n==0:
        return 1
    if n==1:
        return 3
    return (patron3pila(n-1)<<1)+patron3pila(n-2)
```

- De cola

```
'''
Explicacio'n: Calcula el n-simo nu'mero del patro'n:
    1,3,7,17,41,99,239,577,1393...
Dominio: Un nu'mero natural (el valor ma's peque~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron3cola(n):
    if n==0:
        return 1
    if n==1:
        return 3
    return patron3aux(n-2,1,3)
'''
Explicacio'n: Calcula con recursio'n de cola el n-simo nu'mero del
    patro'n: 1,3,7,17,41,99,239,577,1393...
Dominio: 3 nu'meros naturales (el valor ma's pequen~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron3aux(n,x,y):
    if n==0:
        return (y<<1)+x
    return patron3aux(n-1,y,(y<<1)+x)
```

3.4. Patrón 4

3.4.1. Funciones matemáticas

- Función de pila (grado 1):

$$pila(n) = \begin{cases} 2 & si \quad n = 0 \\ pila(n-1) * 2 - (n \text{ mód } 3) + 3 * ((((n-1) \text{ mód } 3) + 1)//3) & sino \end{cases}$$

- Funciones de cola:

$$cola(n) = aux(n, 2, 1)$$

$$aux(n, suma, c) = \begin{cases} suma & si \quad n = 0 \\ aux(n-1, suma * 2 - (c \text{ mód } 3) + 3 * ((((c-1) \text{ mód } 3) + 1)//3), c+1) & sino \end{cases}$$

3.4.2. Código

- De pila

```
'''
Explicacio'n: Calcula con recursio'n de pila el n-simo nu'mero del
    patro'n: 2,3,4,11,21,40,83,165,328...
Dominio: Un nu'mero natural (el valor ma's peque~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron4pila(n):
    if n==0:
        return 2
    return (patron4pila(n-1) << 1)-(n%3)+3*((((n - 1)%3)+1)//3)
```

- De cola

```
'''
Explicacio'n: Calcula el n-simo nu'mero del patro'n:
    2,3,4,11,21,40,83,165,328...
Dominio: Un nu'mero natural (el valor ma's peque~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron4cola(n):
    return patron4aux(n,2,1)
'''
Explicacio'n: Calcula con recursio'n de cola el n-simo nu'mero del
    patro'n: 2,3,4,11,21,40,83,165,328...
Dominio: 3 nu'meros naturales (el valor ma's pequen~o de n es 0).
Codominio: Un nu'mero natural.
'''
def patron4aux(n,suma,c):
    if n==0:
        return suma
    return patron4aux(n-1,(suma << 1)-(c%3)+3*((((c -
        1)%3)+1)//3),c+1)
```

Referencias

[1] E. R. Jiménez, *Fundamentos de Programación.* 2022.